Funkat, Gert; Detschew, Vesselin; Heyn, A.; Kaeding, Anne-Kathrin; Specht, Martin:

Eine konfigurierbare Architektur für die Verwaltung, Verarbeitung und Visualisierung von biologischen Signalen

Zuerst erschienen in:	Biomedizinische Technik = Biomedical Engineering. - Berlin [u.a.] : de Gruyter. - 43 (1998), s3, S. 19-22.
Erstveröffentlichung:	1998
Datum Digitalisierung:	17.07.2009
ISSN (online):	1862-278X
ISSN (print):	0013-5585
DOI:	10.1515/bmte.1998.43.s3.19
[*Zuletzt gesehen:*	31.07.2019]

Eine konfigurierbare Architektur für die Verwaltung, Verarbeitung und Visualisierung von biologischen Signalen

G. Funkat[1], V. Detschew[1], A. Heyn[1], A.-K. Kaeding[1], M. Specht[2]

[1]Institut für Biomedizinische Technik und Informatik, Technische Universität Ilmenau

[2]Klinik für Anästhesiologie und Intensivtherapie, Friedrich-Schiller-Universität Jena

ABSTRAKT

Der Artikel beschreibt eine konfigurierbare Architektur für die Verwaltung, Verarbeitung und Visualisierung von biologischen Signalen. Es wird vorgestellt, wie Verarbeitungsobjekte und Datenobjekte separat verwaltet und in Abhängigkeit von der gewünschten Biosignalverarbeitung konfiguriert werden können.

EINFÜHRUNG

Informationssysteme in der klinischen Routine sind häufig heterogen, inkompatibel und redundant. Damit sind Probleme bei der Verwaltung und Speicherung von Daten sowie dem Einsatz unterschiedlicher Applikationen vorgezeichnet. Im Rahmen eines gemeinsamen Forschungsprojektes mit der Klinik für Intensivmedizin und Anästhesiologie wird bei Einsatz moderner Methoden des Software-Engineering ein experimentelles Stations-Informations-System SIS entwickelt. Dabei ergeben sich eine Reihe von Anforderungen, die man im Sinne einer Wunschliste wie folgt aufführen kann.

- Autonome Komponenten (Daten/Verarbeitung)
- Parallelität und Unabhängigkeit
- Dynamische Rekonfiguration
- Zentrale Wartung
- verschiedene Datenbanken / Filesysteme
- Stabilität und Verteilbarkeit
- Eignung für heterogene Systeme

Diese Anforderungen bestimmten z.T. auch die eingesetzten Methoden und Entwicklungs-werkzeuge.

Im Rahmen des SIS ist ein Teilsystem für die Speicherung, Verarbeitung und Visualisierung von EEG-Signalen entwickelt worden. Das Teilsystem ist von einer allgemeinen Eignung für mehrkanalige bioelektrische Signale ausgehend konzipiert worden. Es ist problemlos für weitere Signalarten erweiterbar.

Bei der Entwicklung des Systems sind zwei Schwerpunkte ausschlaggebend gewesen. Zum einen sollte eine Trennung von Daten, Verarbeitung und Sichten für eine hohe Transparenz und Wartungssicherheit sorgen.

Zum anderen sollte das System zur Laufzeit an die Anforderungen des Anwenders anpaßbar sein. Das bedeutet, daß die konkrete Leistung einer Applikation erst zu dem Zeitpunkt bestimmt werden soll, zu dem auch die Forderung durch den Anwender besteht. Das bedeutet eine hohe Fexibilität und einen sehr geringen Overhead an ungenutzten Ressourcen.

KONZEPTION

Grundlage für das entwickelte Teilsystem ist eine Trennung von Daten, Verarbeitung und Darstellung in jeweils eigenen Komponenten. Die Daten-, Verarbeitungs- und Visualisierungsobjekte haben eine formal beschriebene Systemschnittstelle, über die sie miteinander kommunizieren können. Die Idee ist, in den einzelnen Komponenten nur sehr abgegrenzte Funktionalität (z.B. Datenhaltung, Mittelwertberechnung oder Visualisierung) zu kapseln. Damit erreicht man sehr kompakte Objekte, die erstens bestehende Systemressourcen wenig belasten und zweitens über vorhandene Intra-/Internetze schnell und einfach verteilt oder verfügbar gemacht werden können. Die anwenderspezifische Funktionalität (z.B. eine Mittelwertbildung über einen Datensatz) wird erreicht, indem Daten- und Verarbeitungs-/Visualisierungsobjekte in Metaobjekten gekapselt werden. Die Bildung der Metaobjekte ist abstrahiert in Fig. 1 dargestellt.

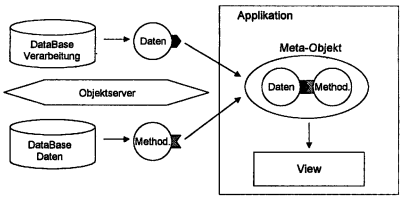

Fig.1: Bildung von Meta-Objekten für eine Applikation

Wesentliche Voraussetzung für den Erfolg des Konzptes ist die formale Beschreibung der Daten-/Applikationsmodelle sowie der Schnittstellen der Objekte. Diese Beschreibung erfolgt mit der Unified Modeling Language UML. Die UML wurde mehrheitlich von Booch, Rumbaugh und Jacobson [2] entwickelt und ist von der Object Management Group als Standard

anerkannt worden. In Fig. 2 sind einige Grundelemente der UML dargestellt.

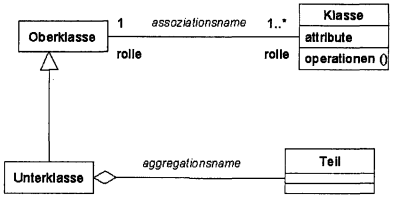

Fig.2: Einige Grundelemente der UML

Die formalen Beschreibungen der Daten-/Applikationsmodelle und der Schnittstellen mit der UML hat eine Reihe von Vorteilen. Sie dienen im Verlauf der Systementwicklung als Plattform, mit deren Hilfe Problemdomänen-Experten und Systementwickler Anforderungen und Gestaltung der Softwaresysteme gemeinsam entwerfen können. Für entwickelte Modelle existieren Werkzeuge, die eine Überführung der Modelle in die gewählte Entwicklungsumgebung automatisieren. Das minimiert Fehler bei der Implementierung. Die Modelle sind weiterhin die Grundlage, notwendige Änderungen oder Erweiterung umsetzen zu können, ohne die Stabilität des gesamten Systems zu gefährden. Damit ist eine exakte Verfolgbarkeit Anforderungsänderungen über die Modelle bis zur Implementierung gesichert.

Für die Daten ist ein objektorientiertes Datenmodell für mehrkanalige Signale entwickelt worden. Dieses Modell ist formal mit der UML repräsentiert. Damit ist eine hohe Stabilität des Datenmodells gegenüber Erweiterungen und vor allem auch Änderungen gesichert. Für die vorgestellte Arbeit ist das Datenmodell für EEG-Signale spezialisiert worden. Das Datenmodell ist problemlos für weitere Formate spezialisierbar. Realisierbar sind auch Datenmodelle, bei denen die Kanäle mit voneinander unabhängigen Samplingfrequenzen abgetastet werden. Für eine ausführliche Darstellung des Datenmodells sei auf die Arbeit von Kaeding et.al. [1] verwiesen. Die Datenobjekte sind von Art und Ort der Speicherung. D.h., die Datenobjekte können physisch in einer Datenbank, in einem Filesystem oder auch in einem Intra-/Internet vorliegen. Für die beschriebene Arbeit gilt die Einschränkung, daß Verarbeitungsobjekte den Ort der Speicherung der Datenobjekte kennen müssen.

Ebenfalls im Ergebnis einer formalen Beschreibung entstehen Datenverarbeitungs-objekte für die verschiedenen klinischen Anforderungen. Die Datenverarbeitungsobjekte für die verschiedenen klinischen Anforderungen sind ebenfalls formal mit der UML beschrieben. Ein Auszug aus dem Applikationsmodell ist in Fig. 2 beschrieben.

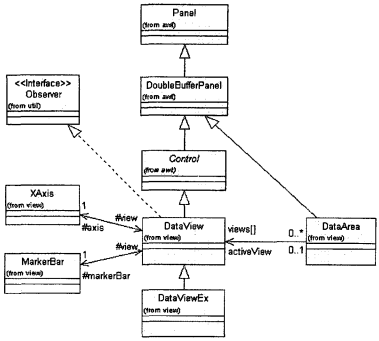

Fig. 3: Formale Beschreibung der Visualisierung

Die Verarbeitungsobjekte kapseln jeweils ein Verfahren zur Biosignalverarbeitung und können für verschiedene mehrkanalige Biosignale konfiguriert werden. Das gilt auch für die Visualisierungsobjekte, d.h. jedem Datensatz wird ein eigenes Visualisierungsobjekt zugeordnet.

Die Verarbeitungsobjekte können analog zu den Datenobjekte in einer Datenbank, einem Filesystem oder einem Intra-/Internet gehalten werden.

Es gibt eine Reihe von Datenverarbeitungs-Methoden, die bereits im Rahmen früherer Projekte implementiert worden sind und in Form von DLL's vorliegen. Dazu zählen z.B. adaptive rekursive Schätzfunktionen wie Mittelwert, Varianz oder Quantilwert zur Deduktion von Instationaritäten in Biosignalen. Um diese Ressourcen nutzen zu können, ist eine Schnittstelle implementiert worden. Die Beschreibung der Schnittstelle wird von der Seite der Applikationsobjekte vorgegeben. Das ist notwendig, um eine hohe Vereinheitlichung der Schnittstellen zwischen Daten- und Verarbeitungsobjekten zu sichern. Es ist dann unter Umständen notwendig, bestehende DLL's mit den überarbeiten Schnittstellen erneut zu übersetzen. Der Aufwand ist vor dem Hintergrund der Wiederverwendung bestehender DLL's zu vertreten.

In der Visualisierungskomponente werden die Daten- und die Verarbeitungsobjekte entprechend der jeweiligen Nutzeranforderungen mit Hilfe der Schnittstellen zu Metaobjekten konfiguriert. Diese Metaobjekte ermitteln die geforderten Ergebnisse und bereiten sie für die Visualisierung auf. Die Orginaldaten werden grundsätzlich nicht verändert. Die in der Visualisierungkomponente erzeugten Metaobjekte können aber Daten ändern, Ergebnisse ableiten oder sie können auch komplett über ein Netz transferiert werden, um an einem anderen Ort visualisiert zu werden.

Die Implementierung des vorgestellten Konzeptes ist in Java erfolgt. Ausschlaggebend waren dafür die folgenden Punkte:

- pure Objektorientierung
- Multithreading
- Garbage Collection
- keine Pointer
- portabel durch Bytecode
- Intra-/Internet-Unterstützung
- Sicherheitsmechanismen

Die pure Objektorientierung ermöglicht es, ohne Paradigmen-Bruch von der Anforderungsanalyse über die Modellierung von Daten und Applikationen bis hin zur Implementierung zu arbeiten. Die einfache Handhabung multipler Threads erlaubt die parallele Verarbeitung z.B. für mehrkanalige Signale. Der Mechanismus der Garbage Collection übernimmt die Verwaltung für nicht referenzierten Speicherraum. Die fehlenden Pointer schließen eine klassische Fehlerquelle z.B. für Speicherlecks aus. Ein großer Vorteil ist das Konzept der Interpretation von Bytecode durch virtuelle Maschinen. Das bedeutet, daß der gleiche Bytecode auf verschiedenen Plattformen lauffähig ist. Virtuelle Maschinen für die verschiedenen Plattformen sind heute z.B. in den meisten aktuellen WWW-Browsern integriert. Die Intra-/Internet-Unterstützung ist die Voraussetzung für eine einfache Umsetzung des Verteilungsaspektes des vorgestellten Konzeptes. Die Sicherheitsmechanismen schützen die Anwender von verteilten Applikationen vor böswilligen Manipulationen.

Ergebnisse

Die vorgestellte Architektur erfüllt eine Reihe unserer Anforderungen. Autonome Komponenten (Daten/Verarbeitung) erlauben, daß mehrere Verarbeitungsprozesse parallel aktiviert werden können. Zur Laufzeit können in Metaobjekten die Daten- und Verarbeitungsobjekte konfiguriert werden. Durch das generelle Konzepte von Java erfolgt die Wartung und Entwicklung zentral. Es sind Schnittstellen für verschiedene Datenbanken und Filesysteme implementiert. Die Verarbeitungs- und Datenobjekte können in heterogenen Systemen verteilt und genutzt werden.

Das eigenständige Datenmodell ist Bestandteil des Gesamt-Datenmodelles. Es ist damit auch einheitliche Grundlage für alle weiteren Zugriffe anderer Applikationen auf die Daten. So werden Datenredundanzen und -inkonsistenzen vermieden.

Der Nutzen der Verarbeitungsobjekte besteht in der Wartbarkeit und der Konfigurationsmöglichkeit. Die Verarbeitungsobjekte sind nicht fest an eine Applikation oder Daten gebunden. Damit kann man sie in einem Intra-/Internet nahezu beliebig organisieren. Das wiederum bedeutet unter anderem, daß die Verarbeitungsobjekte nicht in großer Zahl an die Anwender verteilt werden, sondern nur bei Bedarf von einer bestimmten Stelle geladen werden. Das macht es möglich, daß die Entwickler ihre Implementationen nur an einer bekannten Stelle im Intra-/Internet veröffentlichen und alle Anwender automatisch die aktuelle Version verfügbar haben. Weiterhin können die Verarbeitungsobjekte zusammen mit den Datenobjekten jederzeit in Abhängigkeit der Anwenderforderungen konfiguriert werden.

Die Kombination von Daten- und Verarbeitungsobjekten zu Metaobjekten erlaubt es, festgelegte Konfigurationen an weitere Anwender zu versenden. Solche Anwendungsmöglichkeiten sind sehr vielgestaltig: ein Dienstleister könnte für klinische Anwender bestimmte Konfigurationen (z.B. auf Grund hohen mathematischen Hintergrundwissens) erarbeiten, die nur noch abgearbeitet werden; Arbeitsergebnisse könnten in solchen Konfigurationen zu Dokumentations- oder Demonstrationszwecken versandt werden.

Die separaten Visualisierungsobjekte ermöglichen verschiedene anwenderspezifische Sichten auf Daten und Verarbeitungsergebisse, ohne daß Daten- oder Verarbeitungsobjekte verändert oder angepaßt werden müssen. Außerdem ist so eine ressourcenabhängige Verteilung von Daten (Datenbanken), Verarbeitung (hohe Rechenleistung) und Visualisierung (hohe Grafikanforderung) möglich.

Wertung

Mit der vorgestellten Architektur ist ein stabiles und flexibles Konzept geschaffen worden, das für den weiteren Ausbau offen ist. Durch die Konfigurierbarkeit können Anwenderforderungen weitgehend erfüllt werden. Dazu werden keine neuen Systeme geschaffen, sondern die bestehende Architektur um die geforderten Komponenten erweitert.

Die Daten werden in einem einheitlichen und formal beschriebenen Modell gehalten. Damit wird eine hohe Sicherheit vor Inkonsistenz, Unvollständigkeit, Redundanz erreicht. Eine Änderung oder Erweiterung des Datenmodells ist nachweislich weitgehend unproblematisch.

Die Verteilbarkeit der drei zentralen Komponenten Datenhaltung, Datenverarbeitung und Visualisierung ermöglicht eine physisch unabhängige Lokalisation der Komponenten. Damit können nutzerspezifische Applikationen in einem Intra-/Internet entsprechend der Ressourcen konfiguriert werden.

Als weitere Entwicklungsstufe ist geplant, eine Dienstleistung zu entwickeln und zu implementieren, die es überflüssig macht, daß Verarbeitungs-/Visualisierungsobjekte den Ort der Datenobjekte kennen zu müssen. Damit

könnte eine weitere Automatisierung der Zuordnung von Daten und gewünschter Verarbeitung möglich werden.

Literatur

[1] Kaeding, A.-K. et al.: Objektorientierte Modellierung für mehrkanalige EEG-Signale. Workshop: Anwendung und Möglichkeiten moderner Informationstechnologien in der Biosignalverarbeitung. München, Juli 1998

[2] Rumbaugh, J., Jacobsen, I., Booch, G.: Unified Modeling Language Reference Manual, Addison Wesley Longman, 1997